a year with

mongoDB

a talk by Armin '@mitsuhiko' Ronacher for PyGrunn 2013

That's me.

I do Computers.

Currently at Fireteam / Splash Damage.

We do Internet for Pointy Shooty Games.

let's not beat
around the bush

I don't
like it   :(

but we're not all so negative

“MongoDB is a pretty okay data store”

Jared Hefty (@bridwag)

this is not a rant

it's our experience in a nutshell

we find corner cases

draw your own conclusions

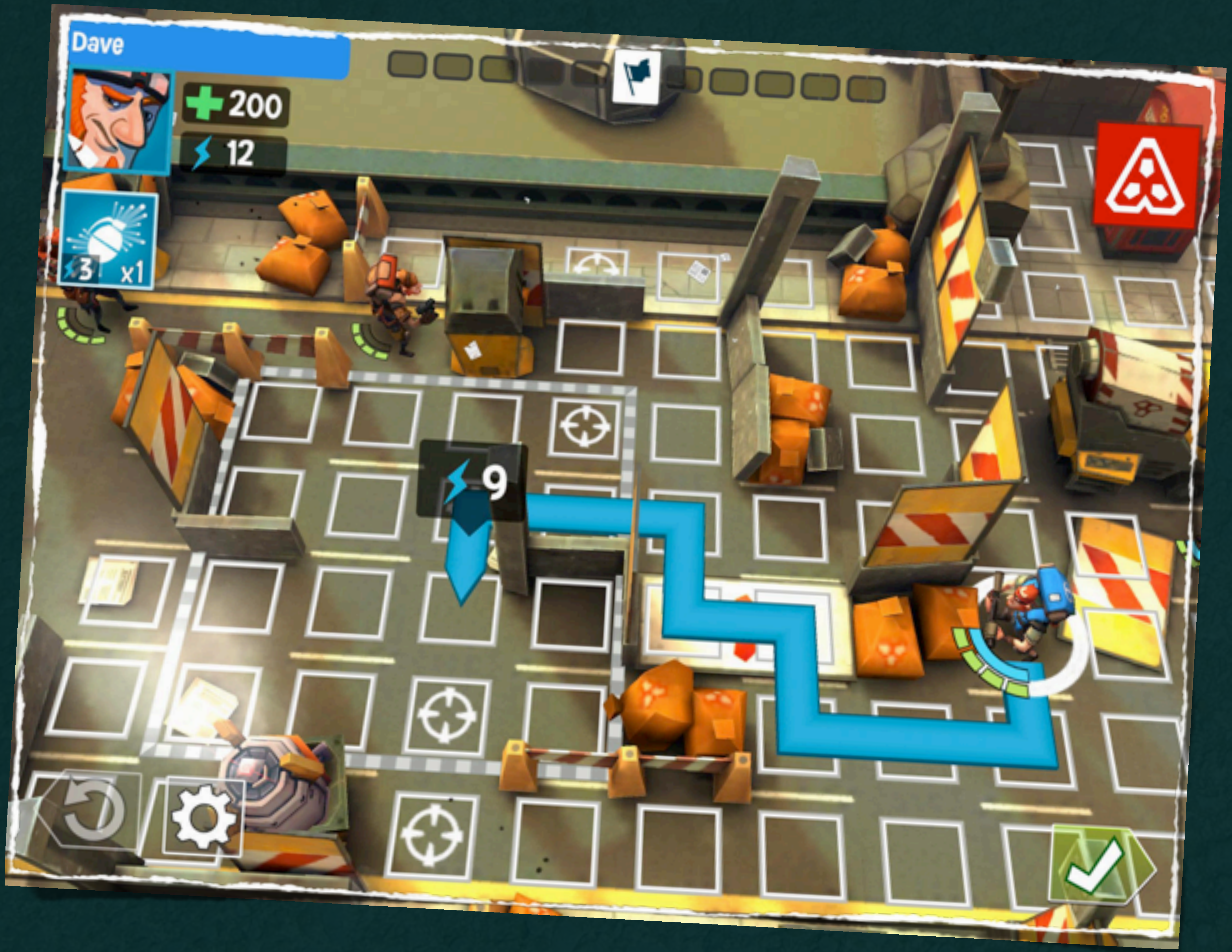
“MongoDB is like a nuclear reactor: ensure proper working conditions and it's perfectly safe and powerful.”

myself on 13th of October 2012

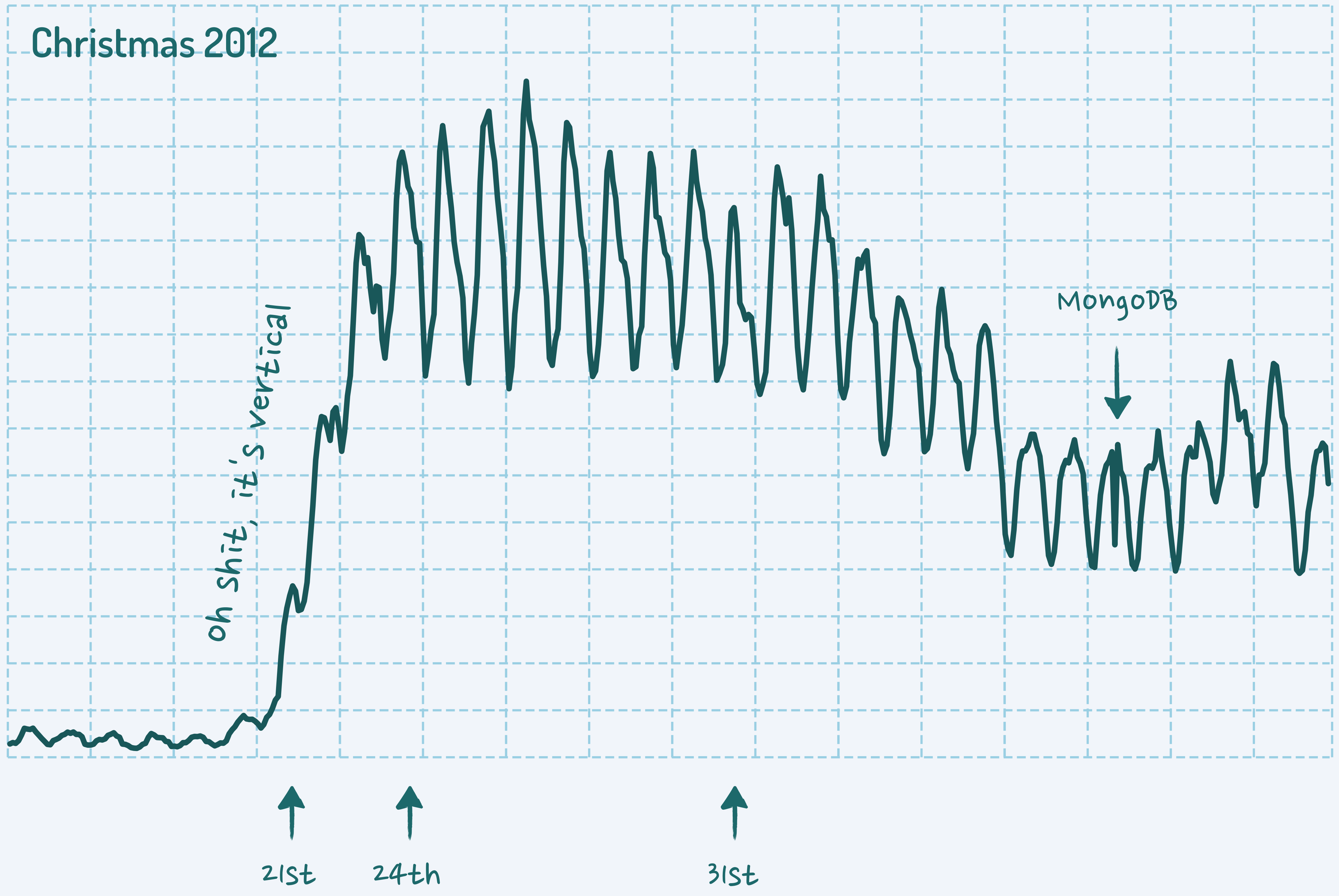
What changed?

RAD Soldiers

RAD Soldiers



RAD Soldiers API calls



?

{ MongoDB Overview }

We recently asked the question

WHY?

Why the fuck did we pick MongoDB?

schemaless

Why the fuck did we pick MongoDB?

schemaless Why the fuck
scalable did we pick
MongoDB?

schemaless
scalable
simple

Why the fuck
did we pick
MongoDB?

schemaless json records

scalable

simple

schemaless	json records
scalable	auto sharding
simple	

schemaless	json records
scalable	auto sharding
simple	think in records

schemaless is wrong

mongodb's sharding is annoying

thinking in records is hard

trololol: two-phase commit

mongod
mongoc
mongos

mongod **mongods**

mongoc

mongos

mongod **mongods**

mongoc **mongocs**

mongos

mongod **mongods**

mongoc **mongocs**

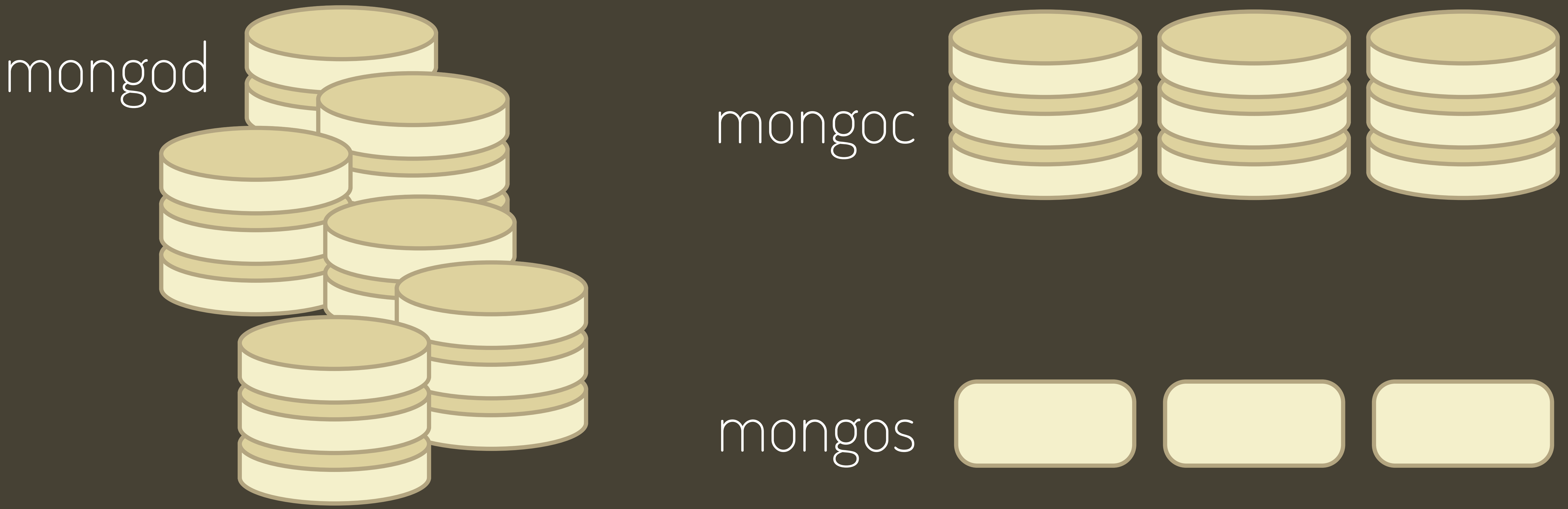
mongos **mongoses**

stores data	mongod	**mongods**
	mongoc	**mongocs**
	mongos	**mongoses**

stores data	mongod	**mongods**
says what's where	mongoc	**mongocs**
	mongos	**mongoses**

stores data	mongod	**mongods**
says what's where	mongoc	**mongocs**
routes and merges	mongos	**mongoses**

Many Moving Parts



{ We Fail }

workers on m1.small

most of the time in IO wait

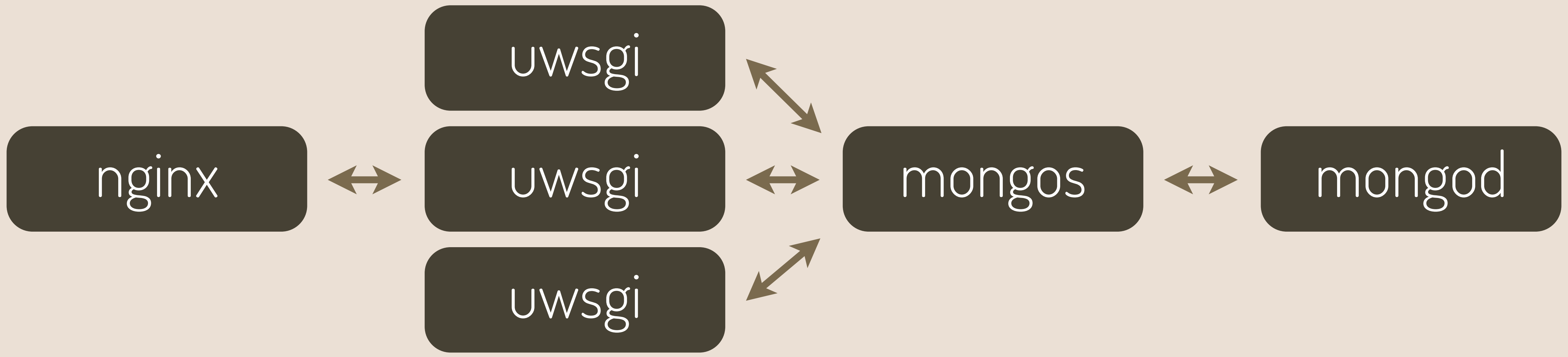
no need for more CPU

oh really?

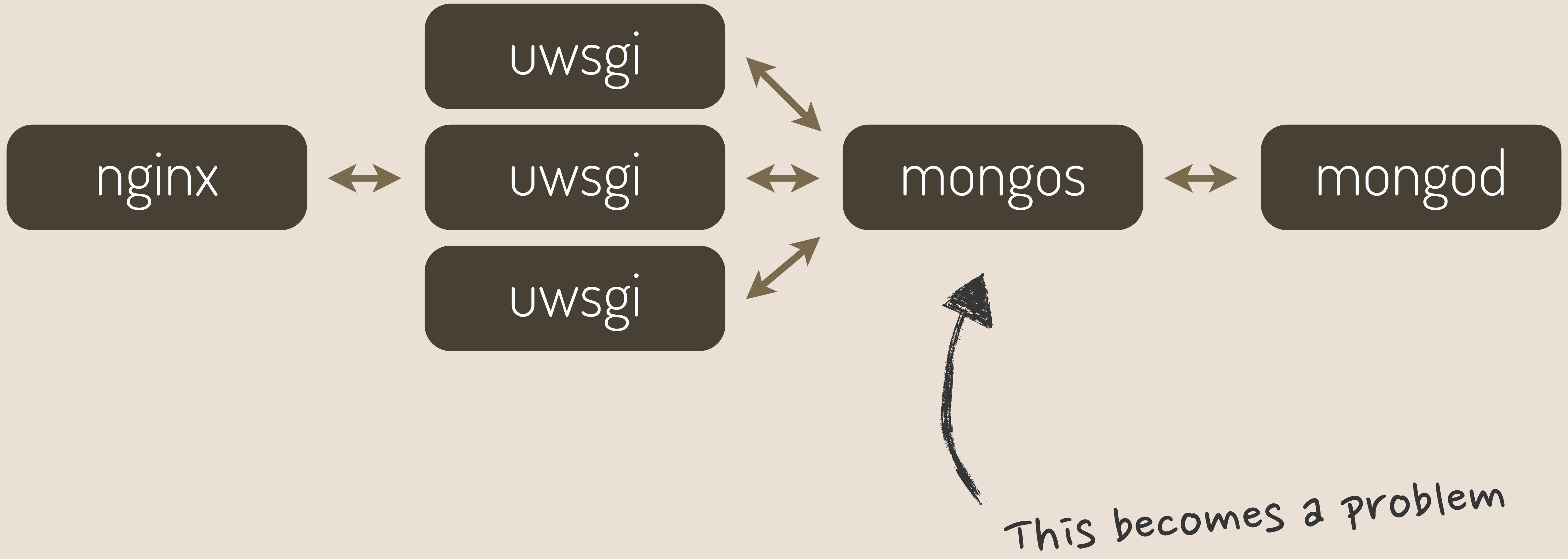
worker setup

$$\begin{tikzcd}\text{nginx} \arrow[r, leftrightarrow] & \text{uwsgi} \arrow[r, leftrightarrow] & \text{mongos} \arrow[r, leftrightarrow] & \text{mongod}\end{tikzcd}$$

worker setup



worker setup



T1 waits for **IO**

T2 uses **CPU**

context switch

worker: mongos, give me data

mongos: mongod, give me data

...

mongos: worker, here is your data

context switch

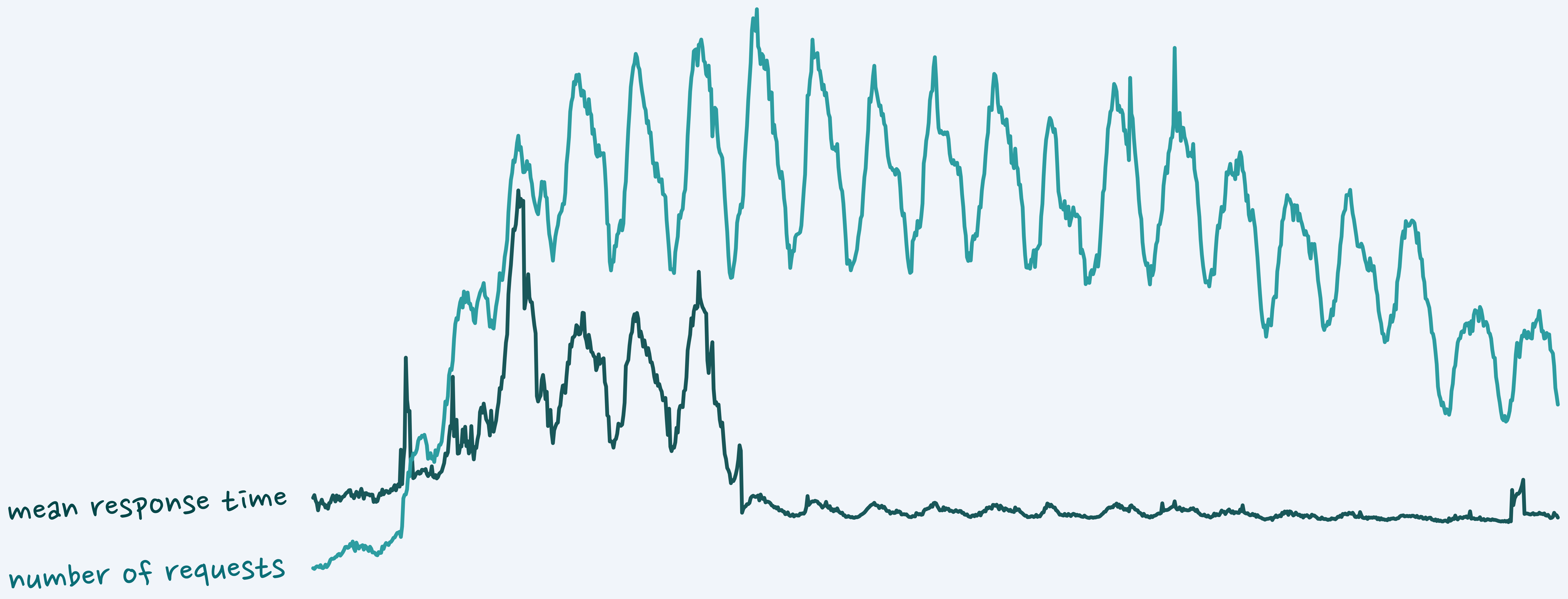
worker: finally! mongos, now give me more data

m1.medium: machines with **2 CPUs***
worker and mongos active at the **same time**
what a novel idea

* cpus[rand() % cpu_choices], might not be an actual CPU

MOAR CPU PLOX

CPU Changes



these are obviously not of the same scale (duh)

EBS

it's pretty bad

Breaking your Instance 101

```
$ dd if=/dev/random of=/var/cache/hah bs=4096 count=1024
```

{ MongoDB's Execution Fails }

No transactions

Document-level Operations

No state

transparent reconnects

NO!

Expectation

- mongos fans out and proxies
- if mongos loses connection worker is good
- voluntary primary election is transparent for worker

Actual Result

- mongos fans out well; technically it's a proxy
- if mongos loses connection it **terminates both sides**
- voluntary primary election **kills all connections**

MongoDB is Stateful

Tail-able Cursors

`getLastError()`

SIGSEGV

Replica Set Annoyances

1. Add Hidden Secondary
2. Witness it synchronizing
3. Take an existing secondary out
4. Actually unregister the secondary
5. Watch the whole cluster re-elect the same primary and kill all active connections

Breaking your Cluster 101

- add new primary
- remove old primary
- don't shutdown old primary
- network partitions and one of them overrides the config of the other in the mongoc

{ MongoDB's Design Fails }

Schemaless

Schema vs Schema-less is just a different version of dynamic typing vs. static typing

Ever since C# and TypeScript:

static typing with an escape hatch to dynamic typing wins

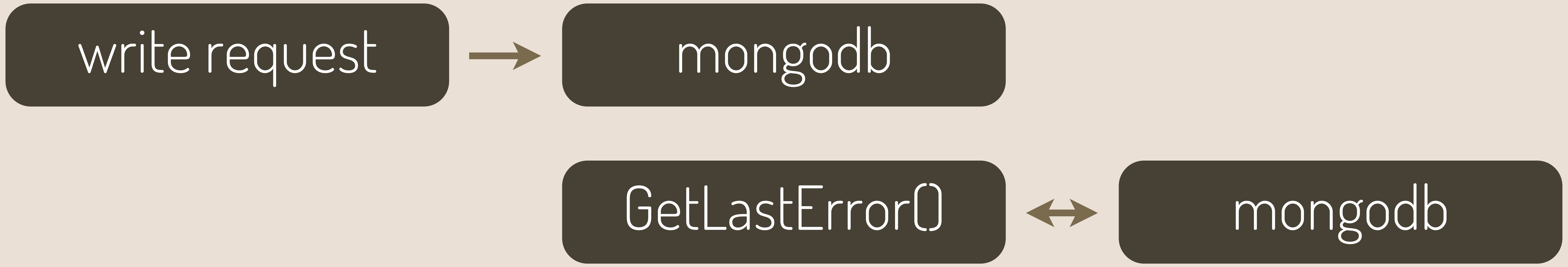
we built an ADT based type system anyways

```
from fireline.schema import types

username = types.String()
profile = types.Dynamic()

x = username.convert('mitsuhiko')
y = profile.convert({'__binary': 'deadbeaf'})
```

GetLastError()

write oddity

why do I need an extra network roundtrip?



performance fun

```python
import os
from pymongo import Connection

safe = os.environ.get('MONGO_SAFE') == '1'
con = Connection()
db = con['wtfmongo']
coll = db['test']
coll.remove()

for x in xrange(50000):
    coll.insert({'foo': 'bar'}, safe=safe)
```

Disappointing

```
$ MONGO_SAFE=0 time python test.py
        1.92 real         1.37 user         0.27 sys

$ MONGO_SAFE=1 time python test.py
        5.57 real         2.50 user         0.62 sys
```

Disappointing

```
$ MONGO_SAFE=0 time python test.py
        1.92 real         1.37 user         0.27 sys

$ MONGO_SAFE=1 time python test.py
        5.57 real         2.50 user         0.62 sys
```

And that's localhost ...

that would not be a problem if safe mode was fast.

As it stands currently safe mode is slower than Postgres

Lack of Joins

(the shitty map reduce is no replacement)

They will happen

1. Before we had joins, we did not have joins
2. not having joins is not a feature
3. I see people joining in their code by hand. Inefficient

RethinkDB has Distributed Joins :-)

```
r \
  .table('marvel') \
  .inner_join(r.table('dc'),
              lambda m, dc: m['strength'] < dc['strength']) \
  .run(conn)
```

MongoDB does **not** have Map-Reduce

(that shitty JavaScript map-reduce thing does not count)

Inconsistent Queries

(and a downright dangerous aggregation query system)

Oh got why!?

```
db.bios.find({
   "awards": {"$elemMatch": {
       "award": "Turing Award",
       "year": { "$gt": 1980 }
   }}
})

db.users.find({"username": "mitsuhiko"})
```

Repeat after me: in-band signalling is **wrong**!

Aggregation Framework comes with SQL Injection

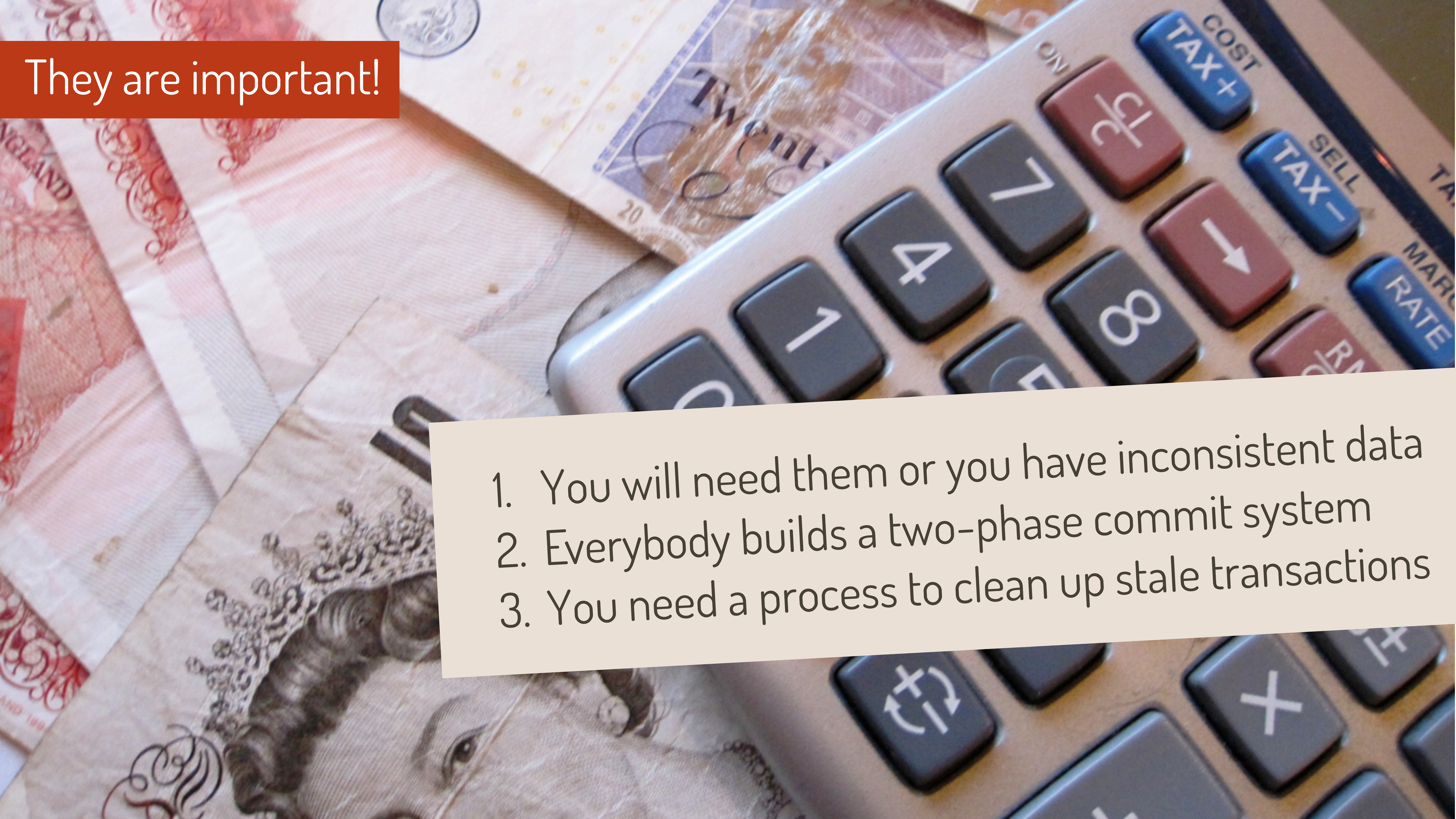
```
db.zipcodes.aggregate({
    "$group": {"_id": "$state",
               "total_pop": {"$sum": "$pop"}}
}, {
    "$match": {"total_pop": {"$gte": 10 * 1000 * 1000}}
})
```

Aggregation Framework comes with SQL Injection

```
db.zipcodes.aggregate({
    "$group": {"_id": "$state",
               "total_pop": {"$sum": "$pop"}}
}, {
    "$match": {"total_pop": {"$gte": 10 * 1000 * 1000}}
})
```

spot the injection :-(

No Transactions



They are important!

1. You will need them or you have inconsistent data
2. Everybody builds a two-phase commit system
3. You need a process to clean up stale transactions

Locks Everywhere

MVCC is good for you

RethinkDB, Postgres and even MySQL support MVCC

Shitty Index Selection

1. MongoDB picks secondary indexes automatically
2. It will also start using sparse indexes
3. It might not give you results back
4. Sometimes forcing ordering makes MongoDB use a compound index

Limited Indexes

1. Given a compound index on [a, b]
2. {a: 1, b: 2} and {$and: [{a: 1}, {b: 2}]} are equivalent
3. Only the former picks up the compound index
4. Negations never use indexes
5. {$or: [...]} is implemented as two parallel queries, both clauses might need separate indexes.

We have a Query Optimizer :P

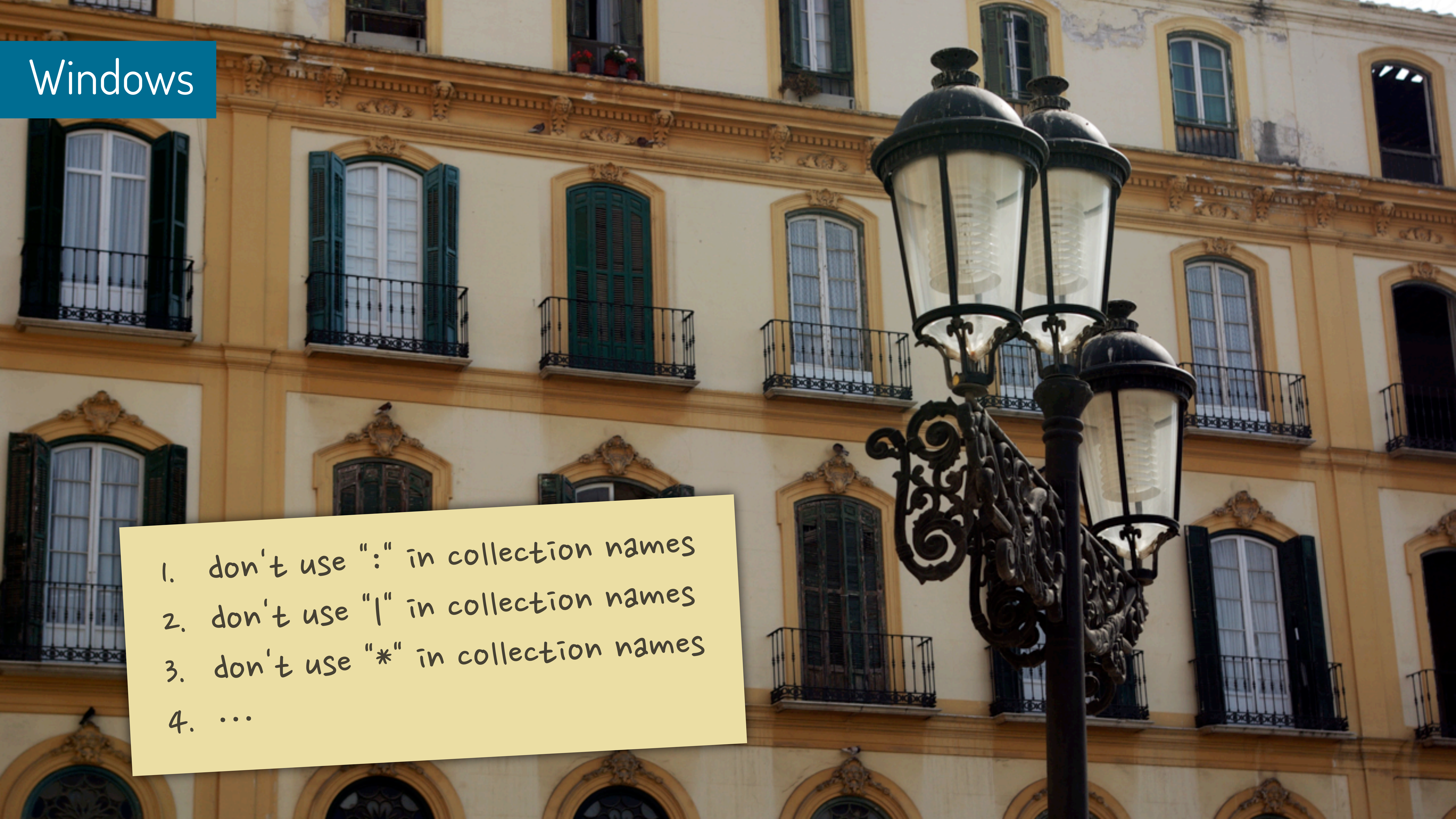
{ Other Things of Note }

Making Mongo not Suck (as much) on OS X

```
$ mongod --noprealloc --smallfiles --nojournal run
```

what are sparse files?

Windows



1. don't use ":" in collection names
2. don't use "|" in collection names
3. don't use "*" in collection names
4. ...

Keys are huge. In our case ⅓ of the Data. Shorten them.

(if only MongoDB had something like a … schema?)

A MongoDB Cluster needs to boot in a certain Order

(Great fun if you have a suspended test infrastructure on Amazon)

MongoDB is a pretty good data dump thing

MongoDB is a pretty good data dump thing

it's not a SQL database

MongoDB is a pretty good data dump thing

it's not a SQL database

but you probably want a SQL database

MongoDB is a pretty good data dump thing

it's not a SQL database

but you probably want a SQL database

at least until RethinkDB is ready

That's it.

Now ask questions.

And add me on twitter: @mitsuhiko

Slides at lucumr.pocoo.org/talks

Legal Shenanigans

Creative Common Sources for Images:
CPU by EssjazNZ: http://www.flickr.com/photos/essjay/4972875711/
Locks by katiejean97: http://www.flickr.com/photos/katiejean97/7036715845/
Money Money Money by Images_of_Money: http://www.flickr.com/photos/59937401@N07/5474168441/
Through any Window by Josep Ma. Rosell: http://www.flickr.com/photos/batega/1354354592/in/photostream/